**Datasets to evaluate autogenerated code**

**1. HumanEval**

* **Description:** The HumanEval dataset is a popular benchmark for evaluating code generation models. It contains programming problems with corresponding unit tests that can be used to verify the correctness of generated solutions.
* **Unit Tests:** Each problem in the HumanEval dataset is accompanied by unit tests that are used to check the functional correctness of the generated code.
* **Usage:** This dataset is widely used in the evaluation of large language models like OpenAI's Codex, which powers GitHub Copilot.

**2. APPS (Automated Programming Progress Standard)**

* **Description:** The APPS dataset consists of a large set of programming problems, ranging from simple to complex, designed to evaluate code generation models. It covers multiple difficulty levels, from introductory to competitive programming challenges.
* **Unit Tests:** APPS includes test cases that serve as unit tests for the generated code. These tests ensure that the code meets the problem requirements and handles edge cases.
* **Usage:** This dataset is used to benchmark the performance of models across a wide range of programming tasks.

**3. CodeXGLUE?**

**4. MBPP (Mostly Basic Python Problems)**

* **Description:** The MBPP dataset is designed to evaluate code generation models on Python programming tasks. It consists of a large number of Python problems with varying levels of difficulty.
* **Unit Tests:** Each problem in the MBPP dataset is accompanied by a set of unit tests that assess the correctness of the generated Python code.
* **Usage:** This dataset is particularly useful for evaluating models that generate Python code, ensuring they produce functionally correct solutions.

**5. SPoC (Student Programming Contest)**

* **Description:** SPoC is a dataset derived from student submissions in programming contests. It contains problems along with multiple solutions, including some that have been tested against unit tests.
* **Unit Tests:** While not all problems in SPoC come with unit tests, a significant portion does, allowing for the evaluation of code correctness.
* **Usage:** SPoC is often used to evaluate the ability of code generation models to handle real-world student programming problems.

**6. CodeContest**

* **Description:** CodeContest is a dataset comprising competitive programming problems, similar to those found on platforms like Codeforces and LeetCode.
* **Unit Tests:** Problems in this dataset come with comprehensive test cases, which act as unit tests to verify the correctness of the generated code.
* **Usage:** It is used to benchmark the performance of code generation models on competitive programming tasks.

**7. MultiPL-E**

<https://huggingface.co/datasets/nuprl/MultiPL-E> = HumanEval + MBPP translated into other programming languages.

**Papers discussing the evaluation of automatically generated code**

**1. Benchmarks and Metrics for Evaluations of Code Generation: A Critical Review**

<https://arxiv.org/pdf/2406.12655v1>

**2. "Evaluating Large Language Models Trained on Code"**

* **Authors:** Mark Chen, Jerry Tworek, Heewoo Jun, et al.
* **Published by:** arXiv, 2021
* **Overview:** This paper presents the evaluation of Codex, a large language model trained on code. The authors use the HumanEval dataset, which includes unit tests, to assess the functional correctness of the code generated by Codex. The paper provides detailed results on the model's performance across various programming tasks and discusses the role of unit tests in validating code generation.
* **Link:** [arXiv:2107.03374](https://arxiv.org/abs/2107.03374)

**3. "** **CodeBERT: A Pre-Trained Model for Programming and Natural Languages "**

* **Link:** [arXiv:2102.04664](https://arxiv.org/abs/2102.04664)

**4. L2CEval: Evaluating Language-to-Code Generation Capabilities of Large Language Models**

<https://ui.adsabs.harvard.edu/abs/2023arXiv230917446N/abstract>

**5. "Program Synthesis with Large Language Models"**

* **Link:** <https://arxiv.org/abs/2108.07732>

**6. Fully Autonomous Programming with Large Language Models**

<https://dl.acm.org/doi/10.1145/3583131.3590481>

**7. "** **SPoC: Search-based Pseudocode to Code "**

<https://arxiv.org/abs/1906.04908>

**8. EvalPlus**

<https://evalplus.github.io/leaderboard.html>

**Code generation leaderboard:**

<https://huggingface.co/spaces/bigcode/bigcode-models-leaderboard>